AFRL-IF-RS-TR-2004-28
Final Technical Report
February 2004

MESO-ADAPTATION BASED ON MODEL ORIENTED REENGINEERING PROCESS FOR HUMAN-COMPUTER INTERFACE (MESOMORPH)

Georgia State University

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. K516

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 074-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503			
1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** FEBRUARY 2004	**3. REPORT TYPE AND DATES COVERED** Final Jul 00 – Jul 03	
4. TITLE AND SUBTITLE MESO-ADAPTATION BASED ON MODEL ORIENTED REENGINEERING PROCESS FOR HUMAN-COMPUTER INTERFACE (MesoMORPH)			**5. FUNDING NUMBERS** C - F30602-00-2-0516 PE - 62302E PR - DASA TA - 00 WU - 15
6. AUTHOR(S) Melody Moore, David Yu, Cen Shi, and Gnan Hoang			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Georgia State University 35 Broad Street Atlanta Georgia 30303-4013			**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency 3701 North Fairfax Drive Arlington Virginia 22203-1714 / AFRL/IFTB 525 Brooks Road Rome New York 13441-4505			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2004-28
11. SUPPLEMENTARY NOTES AFRL Project Engineer: James M. Nagy/IFTB/(315) 330-3173/ James.Nagy@rl.af.mil			
12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.			**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 Words)*

As computing technology increasingly pervades everyday life, demand for more flexible and adaptable user interfaces increases. With advent of assistive technology for people with disabilities, user interfaces for mainstream applications become more unique and specialized. This requires user interfaces to be adapted to fit their context, consisting of the physical environment, user capabilities and activity factors. The MesoMORPH project examined methods to add automation to context-dependent user interfaces. MesoMORPH is a toolset containing gauges that allow design-time metrics related to feature importance and impact of changes according to a specialized context. MesoMORPH provides support for the following capabilities: create optimized applications for a specified set of tasks, adapt for particular user, adapt for a situation and/or adapt for a new device, such as from a desktop to a Palmtop display.

14. SUBJECT TERMS Software Gauge,DASADA, Dynamic Software Assembly, Software Composition, Run-Time Software Adaptation, Automation, Context-Dependent User Interface			**15. NUMBER OF PAGES** 21
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UL

Table of Contents

List of Figures

List of Tables

1 Introduction

Interactive systems are more complex than ever despite research and technology development efforts to make them more accessible and adaptable to a growing and increasingly diverse set of end-users. Interactive applications are often designed for able-bodied users in typical environments such as offices and homes. However, computing technology is pervading so many everyday activities in the military, administration, business, and domestic spheres that people with widely differing cultural and educational backgrounds, work roles and authorities, and physical and cognitive capabilities are called on to interact with computer-based systems. Often these systems are accessed through special-purpose equipment, general-purpose computer peripherals, and hand-held and wearable devices. As these devices become more ubiquitous, the demand for user interfaces adapted for very specialized and unique environments is increasing.

Ideally, interactive systems should be adapted to fit their context of use. *Context* is defined as the physical environment or situation, human user capabilities, and activity scenarios that may affect the design of a system's user interface. For example, an application that is designed for a desktop computer may not be appropriate for a high-distraction environment such as driving a vehicle. Alternatively, small buttons in a user interface may prevent a person with poor control of a mouse from performing selections accurately. In order to accommodate these new situations or human capabilities, the application's user interface must be adapted.

However, performing this adaptation by traditional development methods is very expensive and time consuming. In order to reduce this cost, the MesoMORPH project is examining methods for adding automation to context-dependent user interface adaptation. Our approach includes dynamic reverse engineering techniques to model the interactive components, or *morphology*, of an application. Then, the new user interface context is modeled and applied to the morphology. The resulting modified model is subsequently transformed to implement an adapted user interface. The main approach is to refine abstract user interface models to provide a better fit for an application's environment, users, or usage scenarios.

This report describes the MesoMORPH project, beginning with a description of ontological excavation, and then a set of heuristics for obtaining the morphology from a system to be evolved. We then present a context specification representation, HAS-L, which supports descriptions of user capabilities, activities, and situational factors. The resulting morphology and context model are combined using knowledge base inferencing to produce a new user interface model adapted for the specified context. We incorporate a case study of a real-world user interface as an example to demonstrate context-dependent transformation.

Ontological Excavation

Applications possess and implement a specific "theory of the world" or ontology. Recovering and modeling this ontology may help inform software developers seeking to extend or adapt an application's functionality for its next release. We have developed a method for the black-box reverse engineering or excavation of an application's ontology. The ontology is represented as a semantic network, and graph theoretic measures are used to identify core concepts. Core concepts contribute disproportionately to the structural integrity of the ontology. We present analyses of ontologies excavated from several interactive applications. From a set of several candidate metrics for identifying core concepts we find node between-ness centrality is a good measure of a concept's influence on ontological integrity and that the k-core algorithm may be useful for identifying cohesive subgroups of core features. We conclude by discussing how these analyses can be applied to support application evolution.

MesoMORPH Toolset

MesoMORPH is a process and toolset for evolving software systems for a new context, including environment changes, activity changes, and user abilities (or disabilities). The MesoMORPH toolset includes gauges that allow design-time metrics related to feature importance and the impact of changes according to a specified context. MesoMORPH provides support for the following capabilities:

- Create an optimized application for a specified set of tasks (for example, a fireman and a doctor may be able to use the same disaster recovery system, but need to perform different tasks with the information)
- Adapt for a particular user (a user with physical disabilities, either natural or imposed by the environment - such as a blind user or a user who is operating in darkness)
- Adapt for a situation (using an application while driving a vehicle, adapting to a heads-up display)
- Adapt for a new device (migrate the user interface from a desktop to a Palmtop display)

Benefits

The goals of the MesoMORPH effort were to speed up the software adaptation process for a new context by a significant factor, provide gauges (metrics) to determine the difficulty and feasibility of adapting an application to a particular context, and automatically reorganize and generate a morphology (user interface) that is adapted to a new context. The following sections detail our approach and results.

2 Methods, Assumptions, and Procedures

We established a theoretical framework for MesoMORPH including a process model and definitions for the Ontology (entities and relationships), Morphology (shape of the current user interface) and Teleology (tasks) of an application. We begin with background on Meso-Adaptation.

2.1 Meso-Adaptation

The MesoMORPH project [1] is exploring *Meso-adaptation* as an alternative to traditional software re-engineering or adaptation. Meso-Adaptation is a form of software adaptation falling between the two extremes of macro-adaptation (major re-engineering) and micro-adaptation (run-time tuning). In Meso-adaptation, a change administrator makes changes to a system by configuring COTS/GOTS components that advertise their capabilities after subjecting them to computer-supported analyses of conceptual cohesion, compatibility and coverage. These analyses yield quantitative estimates through MesoMORPH's *gauges*, which are design-time metrics that enable feasibility evaluation for deciding which components can be integrated into existing systems. Three domains of discourse are included in this analysis:

- Ontology is the system's implicit model of the world. System components are compatible to the extent that their ontologies can be merged reliably. MesoMORPH includes a simple but powerful ontology representation *WorldView* (based on *Scenicview* [2]) and a technology *TransPortal* for supporting the derivation of ontology information from black-box interface behavior.
- Context consists of human capabilities, activity scenarios and situational factors in the assumed context of use. In a rapidly changing situation, there is insufficient time for orthodox requirements engineering, but code modifications, however rigorously applied, neglect the human activities and situational factors driving the change and may lead to dangerously unusable systems. MesoMORPH incorporates a representation of human capabilities, activity scenarios and situational factors in a common representation, *HASL*.
- Software architecture includes architectural adaptation wrappers for ontology and context. MesoMORPH includes architectural specifications in terms of both ontology and change factors, and a technology for reengineering existing interfaces. The process includes architecture gauges and automated synthesis.

MesoMORPH defines representations, gauges, and adaptations at each of these levels. The work described in this report details all three levels.

2.2 Reverse Engineering the User Interface Model

The MORPH work showed that application code can be reverse engineered via static analysis to derive a model of the user interface [3]. However code is not always available for user interfaces that need to be quickly adapted. Stroulia et. al demonstrated a viable approach to analyzing an application dynamically to derive a user interface model from its output [5],[6]. Chan et. al combined both static code analysis and dynamic analysis to detect user interface components [7]. These researchers have demonstrated the feasibility of automating black-box dynamic analysis for user interfaces, so we are not currently focusing on automatically deriving morphology. Instead, we focus on the issues and challenges of applying context to a manually derived user interface.

MesoMORPH currently employs a manual black-box reverse engineering technique which can be used to model legacy system user interfaces when the code isn't available. The process consists of two steps: 1) Deriving the morphology and 2) Modeling the morphology. As an example, Figure 1 shows the Microsoft Windows 98 CD player application. The menu items in the top left of the screen are close together and require a drag-and-drop interaction technique to select from a submenu. Also note the small, densely clustered control buttons for play, reverse, fast forward, stop, and pause in the right hand corner of the user interface. An able bodied user who is adept with a mouse typically finds these menu items and icons relatively easy to select. However, a user with a motor disability, or a user who has a mobile computer with a trackball interface may have difficulty with the small size and proximity of the buttons, leading to user interface errors.

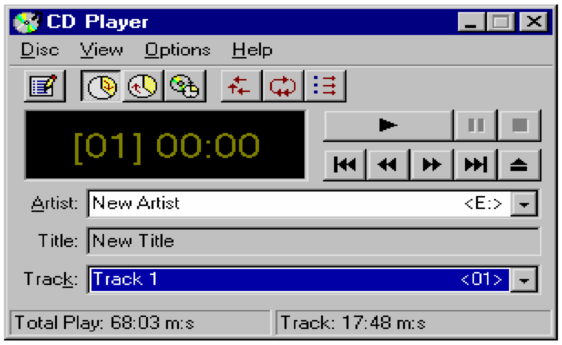


Figure 1*: Windows 98 CD player application*

2.2.1 Deriving the morphology

We model the user interface (UI) by creating a hierarchical *interface map*. This map consists of the UI's visual and semantic grouping components which we term *containers* (e.g. windows, dialog boxes, toolbars), interactive elements or *interactors* (e.g. buttons, text fields, check boxes), and purely presentation components called *information displays*, such as labels or icons. The nodes of an interface map represent these major components of the UI and are also linked using directed edges to show either their point of containment or their point of activation. The hierarchy is rooted by a node representing the application itself; and all menus and controls are represented at the hierarchical level they occur in the interface.

We build this map manually by systematically traversing and activating all the user interface elements in a top-down, left-to-right, and depth-first fashion. These elements and their connections are stored in a Microsoft Visio drawing which represents the entire user interface hierarchy. Figure 2 shows the overall hierarchical interface map for the CD player.

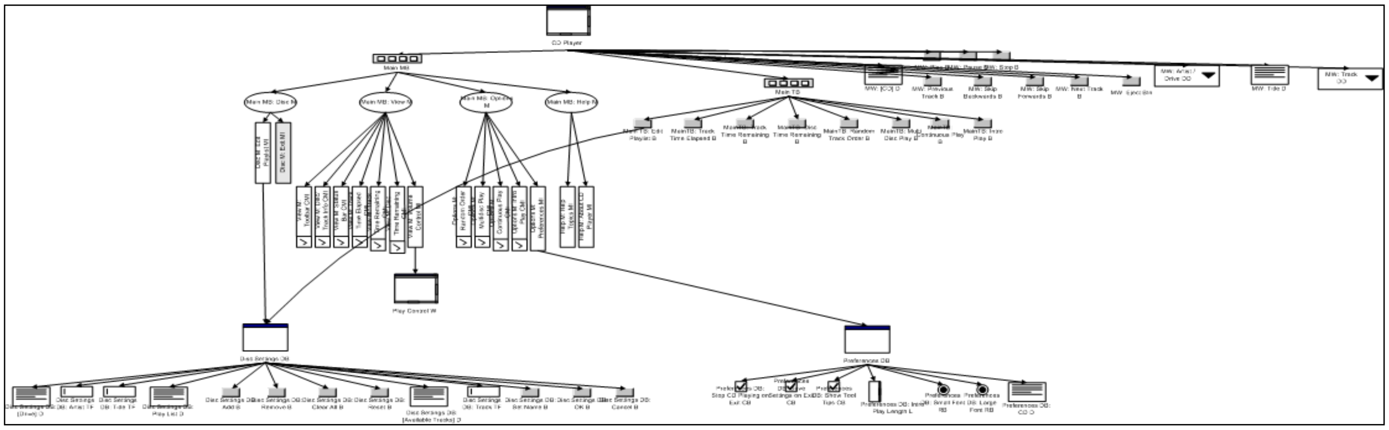


Figure 2***: The overall Windows 98 CD Player Morphology***

2.2.2 Modeling the morphology

The interface map defines the structure and connections of the morphology; but we still must describe the attributes of the individual user interface components. After the interface map is built, we annotate the nodes with abstract component information based on the MORPH concept hierarchy [3]. The functional type of the component, along with specific values for component attributes, is stored in the MS-Visio diagram to create an *annotated interface map.* Figure 3 shows a partial model of the CD interface showing nodes for the CD Player application, Main Menu Bar (MB), the Disc, View, Options, and Help Menus (M), and the two menu items (MI) for the Disc Menu (Edit Playlist and Exit).

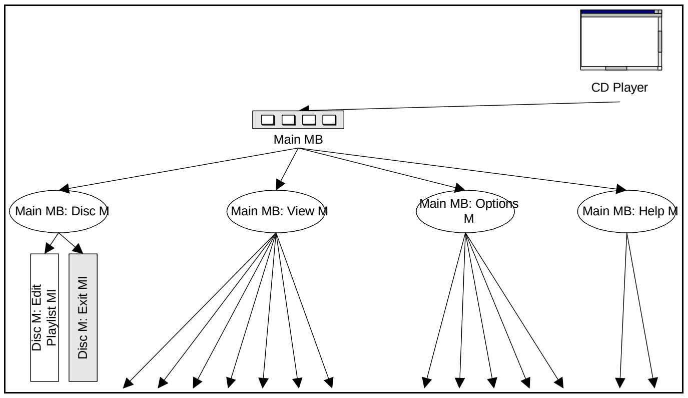
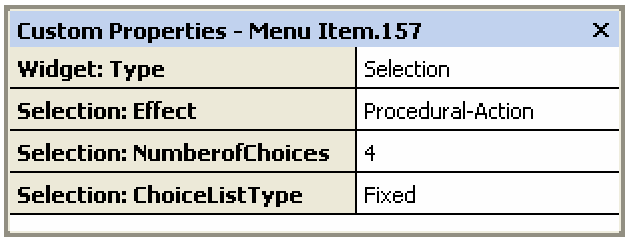


Figure 3***: Portion of the overall morphology of the CD Player***

Figure 4 shows the MORPH-based abstract widget annotations for the main menu, indicating that this is a selection object with a fixed length choice list with four choices, and a procedural action (as opposed to a simple mode switch). This abstract description of the functionality of the widget will assist in choosing an appropriate replacement, if needed, in the new context.



Custom Properties - Menu Item.157

Widget: Type	Selection
Selection: Effect	Procedural-Action
Selection: NumberofChoices	4
Selection: ChoiceListType	Fixed

Figure 4: MORPH widget annotations for main menu

Abstract widget annotations are created for each component of the user interface in the interface map. This information is later used to generate knowledge base representations of the abstract widgets in order to decide which components may need to be changed depending on the context specification.

2.2.3 Representing context

The context of an application includes human, situational, and activity factors that can influence the appropriate design of its user interface. In order to automate the transformation of the interface for a new context, we must be able to describe and represent the context. We have created the Human Activity Situation Language (HAS-L) to fill this requirement.

HAS-L describes context in terms of vectors of capabilities, capability ranges, and capability values. A *capability* is defined as an attribute of a human user or situation that determines the extent to which a task can be performed. A c*apability range* is a set of possible alternatives for a given capability. Lastly, a c*apability value* represents a particular selection for a capability from a capability range for a given context.

Human context factors

The human context of an application may depend on a number of factors, including physical disabilities (such as blindness) or *imposed disabilities* (such as operating in the dark). Because most user interface designs target able-bodied users, these factors can have a profound impact when adapting for a user with a disability. The HAS-L human context model is derived from the information processing model of the general assistive technology system user described by Cook and Hussey [8]. This model divides human capability into several categories, including sensory function, perceptual and cognitive function, psychosocial function, and motor and effector (muscular elements of the body that provide movement) control. For the purposes of our study, we have chosen to represent sensory, motor, and effector capabilities as the human context of an application.

Sensory Function in the HAS-L model includes visual and auditory factors. For example, *visual acuity* refers to the ability to focus on an object, which can be affected by the object's size, contrast with its background, and separation from surrounding objects [8]. A user with low visual acuity may be able to use a graphical interface, but the buttons and icons need to be large, bright, and spaced apart. *Visual field* describes peripheral vision abilities; *visual tracking* refers to the ability to follow a moving object with the eyes. *Visual accommodation* describes the point at which the user's eyes can best focus. We add visual color perception to the model to accommodate users who may be colorblind. Audio amplitude and frequency are determinants of hearing capabilities. The HAS-L sensory factors and their capability ranges are summarized in table 1:

Capability	**Capability Range**		
Visual acuity	None	Reduced	Normal
Visual field	None	Skewed (left or right)	Normal
Visual tracking	None	disjunctive	Normal
Visual accomodation	None	Range (diopters)	Normal
Visual color perception	Colorblind red-green	Colorblind other	Normal
Audio amplitude	None	Reduced	Normal
Audio frequency	Low threshold	High threshold	Normal

Table 1*: Sensory factors for human context*

Motor and effector factors in the HAS-L human context model describe a user's ability to use input devices to communicate with an application. Typically these devices are based on muscle movements, but they may be based on other biometric input such as direct brain interfaces [9]. Fitt's law states that time to move to a target decreases with proximity and larger sized objects, and decreases with distance and smaller objects [10]. Therefore a user with decreased motor abilities may need selection targets to be larger and closer together. Table 2 details the motor and effector factors of HAS-L.

Capability	**Capability Range**		
Speed of cursor movement	Slow	Reduced	Normal
Selection accuracy (resolution)	Low	Reduced	Normal
Reaction time	Slow	Delayed	Normal
Range of movement	Small	Reduced	Normal
Endurance	Low	Reduced	Normal

Table 2*: Motor and effector factors for human context*

Situational context factors

Situational context factors include aspects of the physical environment for the application, such as the screen size of the output device, lighting conditions, or distractions (such as driving a vehicle). The HAS-L situational context model is detailed in table 3:

Capability	**Capability Range**		
Screen Size	Small (Pilot)	Normal (CRT screen)	Large (Smart Board)
Input Device	Continuous (Mouse, Joystick)	Direct (pointing)	Discrete (Switch, keyboard)
Selection Mechanism	Switch	Dwell	
Vision required (location of display)	Central	Peripheral	All
Light	Low	Normal (indoor)	Bright (outdoor)
Sound environment	Normal	Quiet	Noisy
Distraction level	Low	Medium	High

Table 3*: Situational context factors*

Activity context factors

Activity context factors pertain to the user's overall goals for the application, such as:

- Risk level - how important is the activity to the success of the overall mission?
- Reversibility - can the activity be reversed?
- Error margin - how sensitive is the activity to error?
- Cognitive load - decision making, logical reasoning, analysis requirements?

The HAS-L activity factors are detailed in table 4:

Capability	**Capability Range**		
Attention level (of the activity)	High	Medium	Low
Cognitive Load (reasoning req'd)	High	Medium	Low
Error Margin (How sensitive?)	High	Medium	Low
Memory Load (mnemonic demands)	High	Medium	Low
Risk Level (success)	High	Medium	Low
Reversibility (error correct)	High	Medium	Low
Sensory Constraints	High	Medium	Low

Table 4*: Activity context factors*

Deriving context

Obtaining an accurate model of a given context can be a somewhat inexact and subjective process. Ideally, each of the capability ranges within the capability descriptions would be clearly defined and easy to quantify. In reality, determining which capability value is appropriate may depend on thresholds that may be negotiable. For example, how many items must be held in short-term memory to rate a "high" memory load? Currently, we are not addressing these specific questions, rather demonstrating the feasibility of being able to describe context and apply it to a user interface model. The possibilities for creating surveys, clinical diagnostics, and metrics for determining context capability values are significant, but outside the scope of this work.

2.2.4 Transformation

Once the interface map and UI component annotations are complete and the new context has been specified, the final step is to apply the context to the UI model to obtain a transformed UI model. From this new model, a context-specific interface can be generated.

Knowledge representation

To prepare for transformation, the interface map components and the context description are translated into a frame-based knowledge representation called NeoCLASSIC [8], [9]. NeoCLASSIC represents user interface components as *individuals*, or object instances, with capability range attributes specified as *role fillers.* Figure 5 shows the automatically generated NeoCLASSIC representation of the main menu interaction component for the Microsoft CD player in the knowledge base.

```
(cl-create-ind 'Main-Menu
   '(and SELECTION-OBJECT
        (fills action Procedural-Action)
        (fills number-of-states 4)
        (fills variability Fixed)
        (fills grouping Not-Grouped)
    )
)
```

Figure 5*: The NeoCLASSIC representation of the main menu widget attributes*

A NeoCLASSIC individual is created for each user interface component, and then the individuals are all loaded into a knowledge base.

Next, the context description is specified and also translated into the NeoCLASSIC representation. In our case study, we described the context of a user with “locked-in syndrome”, a person who is completely paralyzed and unable to speak. Our user communicates through a direct brain interface that intercepts minute changes in brain signals that can be used as control signals for an external device such as a computer. Figure 6 shows the context capability-value pairs for this user for the Microsoft CD Player:

Human Context

Visual acuity	reduced
Visual field	normal
Visual tracking	Normal
Visual accommodation	Normal
Color perception	Normal
Audio amplitude	Normal
Audio frequency	normal
Speed	slow
Selection accuracy	Low
Reaction time	slow
Range of movement	small
Endurance	low

Situational Context

Screen size	Normal
Input device	Continuous
Selection mechanism	Dwell
Vision required	Central
Light	Indoor
Sound environment	Normal
Distraction level	Low
Stress level	Low

Activity context

Attention level	Low
Cognitive load	Low
Error margin	Low
Memory load	Low
Risk level	Low
Reversibility	Medium
Sensory constraints	medium

Figure 6*: HAS-L context representation for severely disabled user with low selection ability*

Widget nuances

In order to identify appropriate replacement interface components in a specific widget toolkit, the information about the available widgets in the application programming interface (API) also need to be added to the knowledge base, along with the nuances that can be adapted such as size and color. We store these API descriptions as NeoCLASSIC *concepts*, which are class or template definitions. The knowledge base also contains a mapping of effects for the capability values, for example a reduced visual acuity field would be mapped to bright colors. Figure 7 shows a description of a button from the tcl/tk toolkit whose size has been specified as large:

```
(cl-define-concept 'Large-Button
     '(and TK-OBJECT
        (fills effect Procedural-Action)
        (fills number-of-choices 1)
        (fills choice-list-type Fixed)
                     (fills length 100)
        (fills width 33)
       )
```

Figure 7: Representation of button widget in NeoCLASSIC

Transformed model and generated UI

The transformation process consists of a collection of automatically generated NeoCLASSIC queries that test each interface component against the constraints of the context, and produce as output a list of replacement widgets from the specified widget API. When all of the interface map components have been transformed, the resulting model can be used as a specification to build a new graphical user interface (although this can be automated, a manual process is currently employed). Figure 8 shows the interface that result from applying the locked-in user's context to the control button portion of the Microsoft CD player model. The buttons are much larger, spaced farther apart, and are brighter to accommodate the user's reduced selection ability and reduced visual acuity.

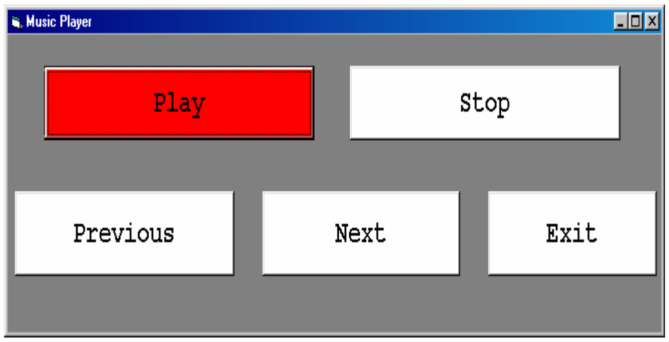


Figure 8: Control portion of evolved user interface for low selection ability

Case Studies

We performed manual experiments with the MesoMORPH process with several pilot applications: iCaster MP3/CD player, Microsoft WinAmp, Yahoo Messenger, and Microsoft Notepad. We completed the iCaster pilot case study described above, modeling the ontology in UML and also in a semantic network and modeling the contextual evolution using the initial HASL specification. We informally performed the gauge analyses to obtain measures of evolvability in the three scenarios, and have produced gauge output such as intensity maps. We composed full models of WinAmp, Yahoo Messenger, and Notepad, and produced the PortL descriptions automatically using the Surveyor tools. We then produced two HASL specifications, one for a user with low attention (driving a car) and one for a user with low selection ability (neural control). We then applied the contexts against the PortL models and generated a new user interface description adapted for the context.

3 Results and Discussion

The initial case study showed that it is possible to automate applying context information to adapt a user interface for a new context. In the course of this work, however, several issues arose:

Dynamic analysis - As with any process involving dynamic analysis, the completeness of the model is an issue. It is very difficult or impossible to generate every possible path through a user interface in order to map the user interface components. In modal interfaces, it may not be possible to reach certain menus without being able to force a particular system state, which in turn may not be possible. Also the halting problem is an issue – how do we know when the interface has been covered? In the MesoMORPH process, we assume that activating all of the menus and icons in depth-first, hierarchical order produces a satisfactory partial model of the user interface.

Ambiguity – It is possible that the MesoMORPH transformation process may identify more than one option for a particular user interface component's implementation. Resolving the ambiguity may require human (system designer) intervention to decide from among the options.

Context conflict - In our case study, we discovered that the adaptations for some capability values may conflict with others. For example, a user with low visual acuity needs icons to be spaced well apart in order to distinguish them visually. However, a user with low cursor movement speed needs icons to be close together in order to be able to target them more effectively. These two capability values are in direct opposition to each other. Again, this situation requires human intervention to make a decision on which of the capability values should take priority in the user interface.

Adding new aspects of context – it is not possible to predict all of the possible capabilities that may need to be accommodated. Others may need to be added as they are discovered. The MesoMORPH HAS-L representation is easily extensible in order to allow new capabilities and capability values to be added.

Context model accuracy - As stated previously, there are many ways of correctly determining context. In order to remove subjectivity from the context descriptions, there need to be set heuristics or procedures for determining context values.

Time consuming - Modeling the morphology manually can be very time consuming. Automating the MesoMORPH blackbox morphology derivation process would represent an enormous time savings.

4 Conclusions

We have demonstrated the feasibility of using automation to streamline the process of adapting software for a particular context. We created a new representation for context, HAS-L, which combines human, situational, and activity factors. We showed in a case study that a real user can be described using HAS-L and that this information can be applied to transform a real-world user interface.

Future work includes incorporating ontological information in order to add centrality (and therefore importance) data to evolutionary procedures. This will allow pruning of interfaces and removal of inappropriate functionality (such as an internet buddy list when adapting the CD player to a driving situation). Because the interface maps are basically graphs with nodes representing components and edges denoting containment or points of traversal, they lend themselves to a network analysis similar to that used in social network theory [13] or urban planning [14]. These analysis methods and graph metrics developed for these two disciplines can be used to measure the morphology's overall efficiency of access and to measure the effects of MesoMORPH's recommended modifications to the interface.

We are also developing similar methods for reverse-engineering the application's domain concepts from the morphology and representing those concepts and their relationships in an ontology that we have chosen to model as a semantic network. Using this representation and the aforementioned metrics, we can identify an application's *core concepts*, concepts that play a significant role in its underlying domain model. Because both representations have the same structural properties, they can be merged into a supergraph where edges connect the morphological elements to the concepts contained in the ontology. In contextual adaptation, an analysis of the relationship between the user interface elements and the core concepts in the ontology can provide metrics about how the modifications are affecting accessibility to those core concepts and recommendations about which interface elements can be removed without loss of integrity to the application's core functions. The ontological information can also affect the presentation of the new user interface components; for example, in the CD player the central concept "play" might result in a play button that is twice as large as other buttons. The eventual goal of MesoMORPH is to support this combined activity of morphological, ontological, and contextual analysis towards a semi-automated adaptation of a system to its new context.

This work can also be vastly expanded in the definitions of context information. Heuristics can be developed to assist in accurately deriving the context capability values. We have currently only addressed physical and sensory capabilities; we have not touched on cognitive disabilities such as memory, recall, and recognition. The MesoMORPH adaptation technology holds much promise for these and other context domains. A full study adapting real-world applications to a variety of contextual factors is underway.

5 References

[1] Moore, M.; Potts, C; Hsi, I; and Yu, D. The MesoMORPH project, www.cis.gsu.edu/~mmoore/MesoMORPH, 2003

[2] Potts, C. ScenIC: A strategy for inquiry-driven requirements determination, Proc. RE'99: International Symposium on Requirements Engineering, Limerick, Ireland. June, 1999

[3] Moore, Melody. *User Interface Reengineering*, Doctoral Thesis, College of Computing, Georgia Institute of Technology, 1998.

[4] Foley, James D., van Dam, Andries, Feiner, Steven K., and Hughes, John F. *Computer Graphics Principles and Practice*, Second Edition, Addison-Wesley Publishing Company, Addison-Wesley Systems Programming Series, 1990.

[5] E. Stroulia and T. Systa. Dynamic analysis for reverse engineering and program understanding, *Applied Computing Reviews*, Spring 2002, ACM Press.

[6] El-Ramly, Mohammad; Stroulia, Eleni; Sorenson, Paul. "Recovering software requirements from system-user interaction traces", Proceedings of the 14th international conference on Software Engineering and Knowledge Engineering, ACM Press, Ischia Italy, 2002.

[7] Chan, Keith; Liang, Zhi Cong Leo; Michail, Amir. "Design Recovery of Interactive Graphical Applications", in *Proceedings of the 2003 International Conference on Software Engineering*, IEEE Press, Portland, Oregon, May 2003.

[8] Peter F. Patel-Schneider, Merryll Abrahams, Lori Alperin Resnick, Deborah L. McGuinness and Alex Borgida. ``NeoClassic Reference Manual: Version 1.0." Artificial Intelligence Principles Research Department, AT&T Bell Labs, 1996.

[9] Lori Alperin Resnick, Peter F. Patel-Schneider, Deborah L. McGuinness, Elia Weixelbaum, Merryll K. Abrahams, Alex Borgida, Ron Brachman, Charles L. Isbell, Kevin C. Zalondek. ``NeoClassic User's Guide: Version 1.0." Artificial Intelligence Principles Research Department, AT&T Bell Labs, 1996.

[10] Cook, A.M., and Hussey, S.M. *Assistive Technologies: Principles and Practice*, *Second Edition*, Mosby Inc., St. Louis, MO, 2002.

[11] Wolpaw, J. R., N. Birbaumer, D.McFarland, G. Pfurtscheller, and T. Vaughan. 2002. Brain-computer interfaces for communication and control. *Clinical Neurophysiology* 113: 767-791.

[12] Fitts, P.M. "The information capabity of the human motor system in controlling the amplitude of movement", *Journal of Experimental Psychology* 48:483-492, 1954.

[13] S. Wasserman and K. Faust, Social Network Analysis. Cambridge University Press, 1994

[14] B. Hillier, Space is the Machine: A Configurational Theory of Architecture. Cambridge, UK: Cambridge University Press, 1996

[15] Hsi and C. Potts, "Studying the Evolution and Enhancement of Software Features," in Proc. Intl. Conf. Software Maintenance, 2000, pp. 143-151.

6 List of Symbols, Abbreviations and Acronyms

COTS – Commercial Off The Shelf (pertains to software)

GOTS – Government Off The Shelf (pertains to software)

HAS-L – Human, Activity, Situational Language (for specifying context)

MesoMORPH – Meso-Adaptation based on Model Oriented Reengineering Process for Human-Computer Interface

MORPH – Model Oriented Reengineering Process for Human-Computer Interface